Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 293 (2016) 1–29

FUZZY sets and systems

www.elsevier.com/locate/fss

A cloned linguistic decision tree controller for real-time path planning in hostile environments ☆

Oliver Turnbull [a,*], Jonathan Lawry [b], Mark Lowenberg [a], Arthur Richards [a]

[a] *Department of Aerospace Engineering, University of Bristol, Bristol, UK*
[b] *Department of Engineering Mathematics, University of Bristol, Bristol, UK*

Received 11 July 2013; received in revised form 13 July 2015; accepted 14 August 2015

Available online 28 August 2015

Abstract

The idea of a Cloned Controller to approximate optimised control algorithms in a real-time environment is introduced. A Cloned Controller is demonstrated using Linguistic Decision Trees (LDTs) to clone a Model Predictive Controller (MPC) based on Mixed Integer Linear Programming (MILP) for Unmanned Aerial Vehicle (UAV) path planning through a hostile environment. Modifications to the LDT algorithm are proposed to account for attributes with circular domains, such as bearings, and discontinuous output functions. The cloned controller is shown to produce near optimal paths whilst significantly reducing the decision period. Further investigation shows that the cloned controller generalises to the multi-obstacle case although this can lead to situations far outside of the training dataset and consequently result in decisions with a high level of uncertainty. A modification to the algorithm to improve the performance in regions of high uncertainty is proposed and shown to further enhance generalisation. The resulting controller combines the high performance of MPC–MILP with the rapid response of an LDT while providing a degree of transparency/interpretability of the decision making.

Keywords: Learning; Fuzzy control; Linguistic modelling; Optimization; Behavioural cloning; Interpretability; UAV

1. Introduction

Behavioural cloning [1–3] has been used to imitate human control of systems that are difficult to model analytically. A limitation of such approaches is clearly that the clone can only be as good as the human it is imitating. An alternative application is in complex real-time systems where the control policy must be determined very rapidly, whilst it may be possible to model such systems, deriving the control policy in real-time is often challenging such as the control of obstacle avoidance for Unmanned Aerial Vehicles (UAVs) [4,5].

☆ This work was supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC), grant number EP/P500109/1.
* Corresponding author.
E-mail addresses: Oliver.Turnbull@bristol.ac.uk (O. Turnbull), J.Lawry@bristol.ac.uk (J. Lawry), M.Lowenberg@bristol.ac.uk (M. Lowenberg), Arthur.Richards@bristol.ac.uk (A. Richards).

http://dx.doi.org/10.1016/j.fss.2015.08.017

UAVs operate in many different environments some of which are highly uncertain, dynamic and, in the case of military applications, often hostile. Current emphasis is on increasing the level of autonomy from pre-programmed path following to autonomous path re-planning to account for unforeseen events [6]. Such re-planning must be completed in real-time and follow some optimised strategy in order to ensure the safety and performance of the vehicle.

An additional requirement on automated path planners if they are to be accepted operationally, is that they must be trusted by human supervisors, i.e. it should be easy to understand the path planners decision making [7,8]. In the context of UAVs, the supervisor may be required to authorize a change to a pre-agreed flight-plan (Management By Consent, MBC) or to provide sufficient information about the decision making process so as to allow the operator to over-ride proposed changes (Management By Exception, MBE) for example in target allocation. The linguistic approach adopted throughout this paper provides an intuitive mechanism by which to convey information regarding the algorithm's decision making, i.e. it is *interpretable*.

There has been much discussion regarding the interpretability of fuzzy systems [9,10], although there remains no agreed definition [11] and consequently it remains a controversial topic [12–14]. Since fuzzy controllers were first proposed by Mamdani [15], the ability to provide an explanation or reasoning for a given decision has been a fundamental driver for using fuzzy sets to model complex systems. Interpretability can be considered as the property that indicates how easily an expert can comprehend the output of the system [13,16]. Although there is no agreed formal definition the following are seen as being characteristic of an interpretable system:

- fewer rules;
- consistency of rules (similar antecedents lead to similar consequents);
- simple rule antecedents containing only a few attributes;
- inclusion of only attributes that are familiar to the user;
- linguistic terms should be intuitively comprehensible (in terms of the membership functions);
- the inference mechanism should provide technically and intuitively correct results.

In light of these multiple criteria it is clear that interpretability is not a simple binary property but a function of the ability to comprehend a system's decision making. Unfortunately, it is generally accepted that there is a trade-off between accuracy and interpretability [12,13] although to further complicate the issue, in some instances over-fitting can lead to rules that are both less interpretable and with lower performance. In this paper we argue that Linguistic Decision Trees offer a good compromise between these two competing notions due to their underlying structure and clear semantic foundations. The method presented in this paper aims to maximise transparency but places slightly greater importance on the algorithm performance. However, it is accepted that alternative methods may offer higher performance at the expense of interpretability.

Path planning with obstacle avoidance is intrinsically NP hard [17,18]. Many solution methods have been studied including optimal control [19,20], potential fields [21,22], graph search [23–25], evolutionary algorithms [26,4] and even manually tuned fuzzy logic controllers [27].

In this paper, Mixed Integer Linear Programming (MILP) [28] is adapted as the reference planner, from which a Linguistic Decision Tree (LDT) will be trained. MILP offers high performance, and since it is used here only for off-line training, its comparatively heavy computational load is not a problem. The MILP planner is implemented within Model Predictive Control (MPC) [29–31] in which the planning problem is repeatedly solved on-line, and only the initial portion of each plan is implemented. This approach introduces feedback, to compensate for uncertainty, and can be proven to be stabilising [32] and to satisfy constraints. The result is a feedback guidance law for the UAV that ensures obstacle avoidance and offers high performance [5]. However, the on-line optimisation limits response speed and provides no justification for the specified output. This motivates cloning.

This paper proposes applying behavioural cloning to an optimized behaviour to combine the benefits of the optimizer's performance with the LDT's rapid decision making. The proposed controller prioritises a rapid decision period suitable for real-time implementation and places a high degree of importance on quality of the decision whilst seeking to maintain interpretability. Specifically the contributions of the paper are:

- an alternative partitioning of bearing attributes to account for circular domains;
- a more generalized Modal-LID3 algorithm (compared to [33]) to account for non-linear output functions;
- Lookahead-LID3 is proposed as a new approach to making predictions with an LDT in areas of high uncertainty.

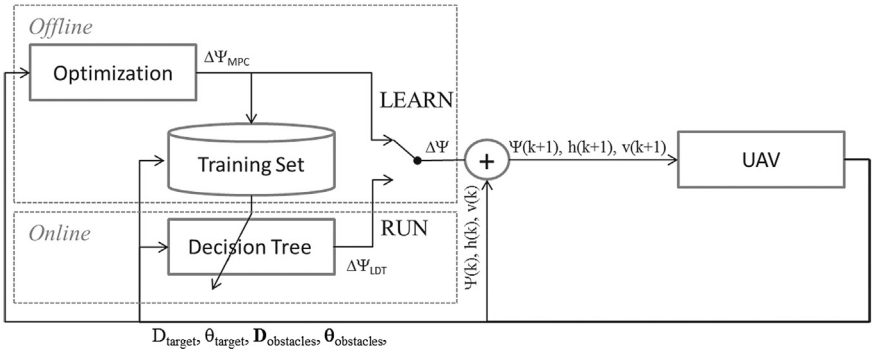


Fig. 1. System architecture for behavioural cloning.

The proposed methods are validated with simulated results but implementation in actual UAVs is beyond the scope of this paper.

The remainder of this paper is organized as follows: Section 2 introduces the relevant background of the applied methods and formally defines the problem statement; Section 3 presents the MPC reference controller used to generate the training data; Section 4 presents several modifications to the standard LID3 algorithm necessary for successful navigation in scenarios containing one or two hostile regions; Section 5 demonstrates successful generalisation to scenarios containing a larger number hostile regions and investigates factors that influence the performance of the cloned controller; Section 6 demonstrates how the semantic nature of the algorithm promotes good interpretability and presents some metrics to measures the interpretability of the proposed controller; Section 7 presents an extension to the algorithm to improve performance in regions of high uncertainty and compares the computational complexity of the new algorithm with that of the optimisation; finally Section 8 highlights the main findings and conclusions of the paper.

2. Preliminaries

2.1. Behavioural cloning

Many high-level human skills, such as decision making, are difficult to imitate. Behavioural Cloning is one technique that has been used to synthesise automatic controllers to perform the same function as human operators [1–3]. The principle is to infer a set of control rules from traces of the operator behaviour.

This paper applies the idea of behavioural cloning to develop an LDT [34,35] controller for path planning in a hostile environment, i.e. navigation to a target obscured by regions of elevated risk, within this paper such regions will subsequently be referred to as "obstacles". It is noted that the obstacles represents soft constraints on the UAV's trajectory, i.e. they *should* rather than *must* be avoided. In order to ensure near optimal trajectories, the training data is generated from an MPC–MILP controller rather than traces of a human operator in contrast to previous instances of behavioural cloning [1–3]. It is intended that the resulting LDT controller can make good approximations to the trajectories from the optimal reference controller, while benefiting from the rapid computation and interpretability of the decision trees.

The architecture of the proposed system is illustrated in Fig. 1. The cloning system has two modes, learn and run. When in the learning mode the model predictive controller is used to generate heading deviation decisions, $\Delta\Psi$, at each time-step, k. For all instances in this paper $\Delta\Psi \in [-65°, 65°]$. The heading deviations are summed with the current heading, Ψ, and used as a reference input to the UAV and recorded with the optimisation inputs (distance and bearing to the target, $D_{target}, \theta_{target}$, and distance and bearing to the obstacles, $\boldsymbol{D}_{obstacles}, \boldsymbol{\theta}_{obstacles}$) in a training set. Once training has been completed the run mode is engaged where the cloned decision tree controller is used to generate the heading deviation commands and performance evaluated. Other inputs to the UAV demanded altitude, $h(k+1)$, and demanded velocity, $v(k+1)$, were fixed at 1 km and 300 ms^{-1}.

Henceforth, the complete system will be referred to as the system or "simulation"; the "UAV" will be referred to as the UAV or "UAV model" which receives inputs from a "trajectory controller". It is important to note that the

trajectory controller provides a heading demand to the UAV which has a lower-level autopilot, capable of translating the heading demand into the required pitch, roll and yaw demands.

The layout of Fig. 1 also illustrates the general concept of a cloned controller. In offline "learn" mode, a complex controller is used to generate a training set. "Complex" here means that the controller offers very good performance but is somehow expensive to deploy online. For example it could be a computationally intensive optimization or a human. That training set is used to develop a decision tree. In online "run" mode, the decision tree is used to approximate the offline controller, seeking similar performance but with less expense. Furthermore, the structure of the decision tree offers easier interpretation of behaviour than the offline control would typically allow.

Since the learning phase employs simulation instead of experiment, it is important to consider if the cloning is representative of real-life results. The only part of the system we are cloning is the controller, i.e. some software, which would present the same outputs, given the same inputs, in simulation as in reality. The relevant question then is whether or not the simulated training generates cases that are representative of those that would be seen in reality. To address these issues Section 5 investigates the ability of the LDT to generalise to new instances while the method proposed in Section 7 further improves the robustness of the algorithms in regions of high uncertainty. Further work is necessary to investigate the sensitivity of the training process to the parameters of the simulator. Possible future work in this direction would be the inclusion of noise within the simulation especially with regard to the UAV sensors.

2.2. Linguistic decision trees

Decision trees were initially applied to classification problems with discrete attributes such as Quinlan's ID3 algorithm [36]. The C4.5 algorithm [37] extended the range of problems decision trees could solve by proposing a crisp partitioning of continuous attributes. This method of partitioning led to generalisation and stability problems due to the sudden switching between partitions for small changes in the inputs. Several fuzzy decision tree methods were proposed [38–41] to overcome the problems associated with crisp sets but these methods are still limited to classification problems. Breiman et al. proposed the CART algorithm [42] to allow decision trees to be applied to regression problems. Another approach to regression problems was proposed by Lawry et al. which combined Quinlan's ID3 algorithm [36] with label semantics [43]. The resulting algorithm, LID3 [44,45], has been used to tackle regression problems such as flood and storm surge prediction [46,47].

The interpretability of decision tree algorithms has previously been assessed as high [16] based on properties such as having a consistent rule base with simple and easily comprehensible rules; the rules typically include only a subset of the input variables; the associated membership functions are intuitive to the user (as they must be predefined and are not modified by the learning process). It is acknowledged, however, that the LID3 algorithm introduces some additional features that reduce the interpretability of the resulting decision tree. There are two significant features of LID3 that affect interpretability: the first is that, unlike a standard ID3 decision tree, each branch or rule has an associated "belief" (although only a few branches will be non-zero for any instance); second is the introduction of a probability distribution on the output variable given each branch, this effect could be mitigated by presenting the user with the de-fuzzified output associated with the distribution. Section 6 demonstrates that despite these restrictions, the rules generated by an LDT remain meaningful and useful to experts.

To summarise, LID3 has the following advantages:

1. It offers an inherent model of uncertainty [35];
2. The model offers good performance, robustness and generalisation [34]; comparable to other regression algorithms such as Naive-Bayes [48], Neural Networks [49] and Support Vector Machines [44,50];
3. The branches of the tree can be interpreted as a set of linguistic rules based on label semantics, thereby improving the interpretability of the tree [16];
4. The linguistic structure of the tree enables linguistic queries or explanations and information fusion [51,52].

This paper builds on the extensive previous work on decision trees [53–57] and takes advantage of their ability to generate accurate models based on training data to offer a real-time approximation to a large scale optimal controller for the complex UAV control problem of path planning subject to additional logical constraints. An additional benefit of the decision tree framework is that it offers a transparent rule-based representation allowing for the easy analysis of the rationale behind real-time decisions. To define terminology used throughout this paper and enable a comprehensive

definition of the contributions of this paper, the original LID3 method [44] to construct Linguistic Decision Trees is now briefly revised.

2.2.1. Label semantics

Proposed by Lawry [43], Label Semantics is a framework for modelling with linguistic expressions based on labels such as *left, ahead, right* or *very close, close, distant, very distant*. The theoretical foundations lie in random set theory and have a strong probabilistic underpinning, providing clear operational semantics for the decision rules generated and providing a representational framework for computing and modelling with linguistic rules.

The rules are formulated from sets of labels, $LA = \{L_1, \ldots, L_n\}$, describing the universe, Ω, of each attribute that provides information about the current state.

Let D_x^I denote the subset of LA that an individual, I, identifies as a suitable description of x. If I varies across a population V with prior distribution P_V, then D_x^I will also vary and generate a random set, denoted D_x into the power set of LA. The result of evaluating the probability of a particular subset of labels, S, for D_x across the population is a distribution on D_x referred to as a mass assignment (see [58] for details of Mass Assignment theory). D_x can be viewed as a description of x in terms of the labels in LA [44], or formally:

Definition 1 *(Label description).* For $x \in \Omega$ the label description of x is a random set from V into the power set of LA, denoted D_x, with associated distribution m_x which is given by:

$$\forall S \subseteq LA, \quad m_x(S) = P_V(\{I \in V | D_x^I = S\}) \tag{1}$$

where $m_x(S)$ is the mass associated with a set of labels S, and:

$$\sum_{S \subseteq LA} m_x(S) = 1 \tag{2}$$

The labels can be modelled as fuzzy sets with associated memberships, or appropriateness degrees (μ), for different values of x where $\mu_L(x)$ reflects the proportion of voters from V who deem L an appropriate label for x. The term 'appropriateness degree' is used partly as it more accurately reflects the underlying semantics and partly to highlight the quite distinct calculus based on this framework.

Definition 2 *(Appropriateness degrees).*

$$\forall x \in \Omega, \ \forall L \in LA \quad \mu_L(x) = \sum_{S \subseteq LA : L \in S} m_x(S) \tag{3}$$

In order to avoid allocating mass to the empty set a full fuzzy covering is assumed such that, for any data element, there always exists a particular label that all voters agree is appropriate regardless of their opinion of other labels.

Definition 3 *(Full fuzzy covering).* A full fuzzy covering of the continuous discourse Ω by LA is such that:

$$\forall x \in \Omega, \ \exists L \in LA \quad \mu_L(x) = 1 \tag{4}$$

Fig. 2 shows a fully fuzzy covering of an attribute describing the distance to an object covered by three trapezoidal fuzzy sets with a 50% overlap, i.e. every value of x is covered by exactly 2 fuzzy sets. From the figure it is clear that $\mu_{Nearby}(40) = 1$ and $\mu_{Distant}(40) = 0.8$. N_F fuzzy sets with a 50% overlap are used throughout this paper. This ensures that only two fuzzy sets overlap and that the appropriateness degrees satisfy:

$$\forall x \in \Omega, \ \exists i \in \{1, \ldots, N_{F-1}\}$$

such that:

$$\mu_{L_i}(x) = \alpha \tag{5}$$

$$\mu_{L_{i+1}}(x) = \beta \tag{6}$$

$$\mu_{L_j}(x) = 0 \ \text{ for } j \notin \{i, i+1\} \tag{7}$$

$$max(\alpha, \beta) = 1 \tag{8}$$

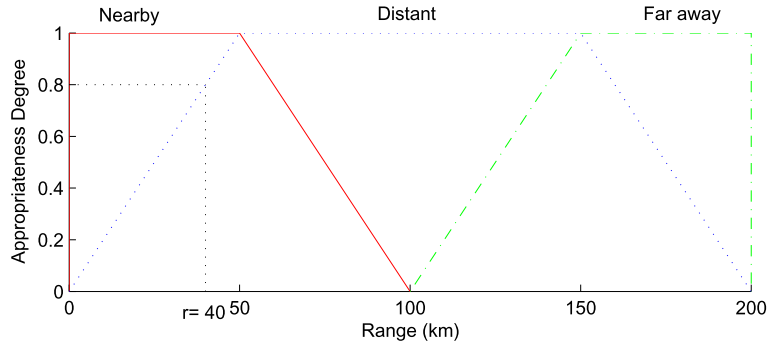


Fig. 2. Fuzzy set coverage of a linear domain.

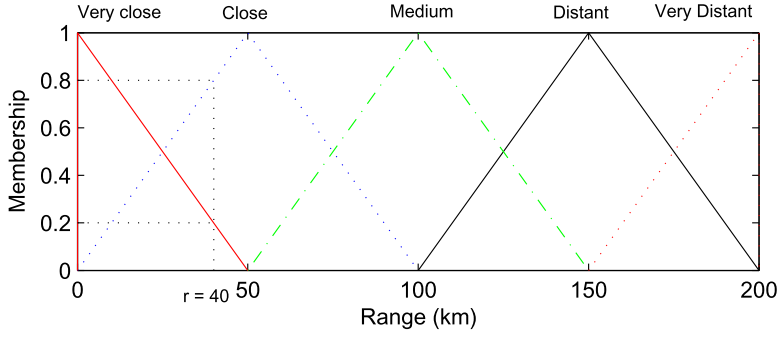


Fig. 3. Focal element coverage of a linear domain.

Under the full fuzzy covering assumption, then w.l.o.g. if it is assumed that $\alpha = 1$ then m_x can be found as follows:

$$m_x = \{L_i\} : 1 - \beta \tag{9}$$

$$\{L_i, L_{i+1}\} : \beta \tag{10}$$

$$\{L_j\} : 0 \quad \text{for } j \notin \{i, i+1\} \tag{11}$$

This mapping enables us to define labels in terms of their appropriateness measures and then infer the underlying mass function directly.

Intuitively, some sets of labels are nonsensical and cannot logically exists, e.g. {*very close*, *very distant*}. The sets of labels that a voter could simultaneously describe as appropriate labels for a single value of x are referred to as *focal sets* or *focal elements*:

Definition 4 *(Focal elements).* The set of focal elements for *LA* is defined by:

$$\mathcal{F} = \{S \subseteq LA | \exists\, x \in \Omega,\ m_x(S) > 0\} \tag{12}$$

Fig. 3 shows the focal elements of the range attribute shown in Fig. 2 where the focal elements have been given more meaningful names, e.g. *Very Close* $\equiv$ {*Nearby*} and *Close* $\equiv$ {*Nearby*, *Distant*}. The resulting membership functions can be considered as a strong fuzzy partition [59]. Therefore, from Eqn. (9) and the appropriateness degrees calculated previously, the following mass assignment can be derived:

$$m_{40} = \textit{Very Close} = \{\textit{Nearby}\} \quad : 1 - 0.8 = 0.2 \tag{13}$$

$$\textit{Close} = \{\textit{Nearby}, \textit{Distant}\} \quad : 0.8 \tag{14}$$

2.2.2. Linguistic ID3

This section reviews the method proposed by Qin and Lawry [44] for learning an LDT from data.

Linguistic ID3 is the algorithm for constructing the linguistic decision tree from a given database of examples, $\mathcal{D} = \{\langle \mathbf{x}(i), x_t(i)\rangle | i = 1, \ldots, N\}$ where $\mathbf{x}(i) = \langle x_1(i), \ldots, x_n(i)\rangle$ is a vector of the potential descriptive attributes $x_1, \ldots, x_n$ and x_t is the target attribute. For compactness $\mathbf{x}(i)$ will subsequently be denoted $\mathbf{x}_i$. Each attribute is covered by a set of focal elements, $\mathcal{F}_1, \ldots, \mathcal{F}_n$ and $\mathcal{F}_t$, consequently a branch can be defined as $B = \langle F_1, \ldots, F_{|B|}\rangle$ where $F_i \in \mathcal{F}_i$ for $i = 1, \ldots, |B|$ and $|B|$ is the depth of branch B. Therefore, the formal representation of the decision tree that is created can be expressed as:

$$\begin{aligned} LDT = \{&\langle B_1, P(F_t^1|B_1), \ldots, P(F_t^{|\mathcal{F}_t|}|B_1)\rangle, \\ &\vdots \\ &\langle B_s, P(F_t^1|B_s), \ldots, P(F_t^{|\mathcal{F}_t|}|B_s)\rangle\} \end{aligned} \tag{15}$$

An information based heuristic guides attribute selection at each node as in ID3 [36] but the information measurements are adjusted in accordance with label semantics. The measure of information defined for a branch B, can be viewed as an extension of the entropy measure used in ID3.

Definition 5 *(Branch entropy).* The entropy of branch B is given by:

$$BE(B) = -\sum_{j=1}^{|\mathcal{F}_t|} P(F_t^j|B) \log_2(P(F_t^j|B)) \tag{16}$$

The probability of F_t^j given B can be found as follows:

$$P(F_t^j|B) = \frac{\sum_{i=1}^N \xi_i^j P(B|\mathbf{x}_i)}{\sum_{i=1}^N P(B|\mathbf{x}_i)} \tag{17}$$

where ξ_i^j is the degree to which $\mathbf{x}_i$ belongs to target focal element F_t^j:

$$\xi_i^j = m_{x_t(i)}(F_t^j) \tag{18}$$

and the probability of a branch given $\mathbf{x}$ is the product of the membership of $\mathbf{x}$ to each focal element along the branch:

$$P(B|\mathbf{x}) = \prod_{r=1}^{|B|} m_{x_r}(F_r) \tag{19}$$

In the case that $P(B|\mathbf{x}(i)) = 0$ for all $i = 1, \ldots, N$, i.e. none of the training data give non-zero probability to branch B, then we allocate uniform probabilities to each focal element given B:

$$P(F_t^j|B) = \frac{1}{|\mathcal{F}_t|} \quad \text{for } u = 1, \ldots, |\mathcal{F}_t| \tag{20}$$

If branch B is now to be expanded, each free attribute is evaluated in turn based on the expected entropy which is defined as follows.

Definition 6 *(Expected entropy).*

$$EE(B, x_j) = \sum_{F_j \in \mathcal{F}_j} BE(B \cup F_j) \cdot P(F_j|B) \tag{21}$$

where $B \cup F_j$ represents the new branch obtained by appending the focal element F_j to the end of branch B.

The probability of F_j given B can be calculated as follows:

$$P(F_j|B) = \frac{\sum_{i \in \mathcal{D}} P(B \cup F_j|\mathbf{x}_i)}{\sum_{i \in \mathcal{D}} P(B|\mathbf{x}_i)} \tag{22}$$

The *Information Gain* (*IG*) obtained by expanding branch B with attribute x_j can now be defined as:

$$IG(B, x_j) = BE(B) - EE(B, x_j) \tag{23}$$

The aim of tree structured learning methods is to partition the data such that each sub-region is "purer" in terms of the mixture of class labels than the un-partitioned set. A tree is generated by first selecting the most informative attribute and then expanding a branch of the tree for each focal element on that attribute; this process is repeated using any of the remaining attributes until the maximum specified depth is reached or some other criterion met.

Once the tree has been constructed, new instances of the descriptive attributes can be used to make predictions. Given a new instance $\mathbf{x}$, the predicted value, $\hat{x}_t$ is found as follows:

$$\hat{x}_t = \sum_j P(F_t^j|\mathbf{x}) E(x_t|F_t^j) \tag{24}$$

It is noted that this expression is adapted in Section 4.2 due to the nature of the heading deviation function that is being approximated. According to Jeffrey's rule [17], for a decision tree consisting of s branches:

$$P(F_t^j|\mathbf{x}) = \sum_{v=1}^{s} P(F_t^j|B_v) P(B_v|\mathbf{x}) \tag{25}$$

and:

$$E(X_t|F_t^j) = \int_{\Omega_t} x_t\, p(x_t|F_t^j)\, dx_t \tag{26}$$

$$= \frac{\int_{\Omega_t} x_t\, m_{x_t}(F_t^j)\, dx_t}{\int_{\Omega_t} m_{x_t}(F_t^j)\, dx_t} \tag{27}$$

If a data element exceeds the range of the training data $[R_{min}, R_{max}]$ for a particular attribute when using the decision tree to make predictions, it is temporarily assigned the value of R_{min} or R_{max} depending on which side of the range the element appears.

Having outlined the method to be applies, the problem to be solved is now formally defined in the next section.

2.3. Problem statement

The central problem investigated in this paper is as follows: control a UAV to reach a target obscured by one or more obstacles by following a near optimal trajectory, minimised with respect to path duration, and subject to constraints representing the vehicle dynamics. The path should remain outside known obstacle regions at all times. The obstacle size and shape is assumed to be constant in all scenarios and the obstacles and target are assumed to be stationary.

The problem is explored using a simulation provided by the Group for Aeronautical Research and Technology in Europe, Flight Mechanics Action Group 14 (GARTEUR, FM AG14) [60]. The scenario addressed by the simulation includes the problem of generating a trajectory for a UAV to reach a target obscured by multiple Surface to Air Missile (SAM) sites. The SAM sites, or obstacles, are modelled as circular regions of risk which, which it is desirable to remain outside of, and are consequently a soft constraint on the UAV trajectory. In some instances, it may even be necessary to pass through one or more obstacle regions in order to generate a feasible trajectory to the target. If it is necessary to intrude into an obstacle region then it is desirable to keep the intrusion as small as possible as the risk is inversely proportional to the distance to the obstacle centre. When considering multiple obstacles it is therefore sensible to minimise the total risk rather than any other metric such as the number of obstacles entered.

The simulation includes a 3-dimensional UAV model, multiple SAM sites and a target that the UAV is trying to reach. For the purposes of this work, the UAV is assumed to fly at constant altitude and to be equipped with an autopilot enabling it to follow a supplied reference heading. A simple unicycle model [61] is used to approximate the UAV dynamics in the optimisation.

The simulation provides the distance and bearing to the target $(D_{target}, \theta_{target})$ and any detected obstacles $(\boldsymbol{D}_{obstacles}, \boldsymbol{\theta}_{obstacles})$ as outputs where all angles are relative to the UAV's current position and heading as illustrated in Fig. 4.

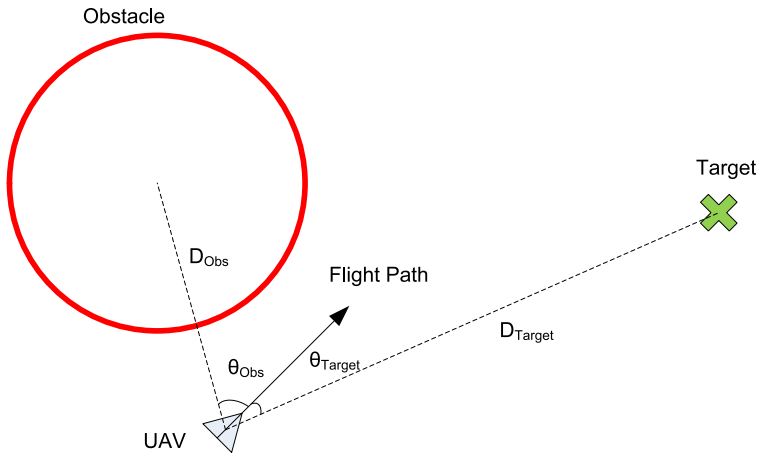


Fig. 4. Example scenario illustrating range and bearing definitions.

The required obstacle avoidance controller is therefore of the form:

$$\Delta\Psi = f(D_{target}, \theta_{target}, \boldsymbol{D}_{obstacles}, \boldsymbol{\theta}_{obstacles}) \tag{28}$$

Predicting a heading deviation and choosing the flight path axes as the frame of reference results in a data representation that is invariant under global translation and rotation. The independence with respect to these basic transformations is beneficial as it effectively reduces the problem space and improves generalisation by allowing many different scenarios to be mapped on to a single representation.

3. MPC–MILP reference controller

This section reviews the method from [5] for UAV guidance using online MILP.

The optimisation below solves for the minimum time path to the target using a linear approximation [62–64] to the UAV dynamics. Variables are: time to target N_k; acceleration $\mathbf{a}(k)$, for predicted time-steps $k \in 0, \ldots, (N_k - 1)$; obstacle intrusion, $\mathbf{c}(k, s)$, for time-steps $k \in 0, \ldots, (N_k - 1)$ and obstacles $s \in 0, \ldots, N_o$ where N_o is the total number of obstacles.

The optimisation is formulated as follows:

$$\min_{N_k, \mathbf{a}(k), \mathbf{c}(k,s)} N_k + \gamma \sum_{k=0}^{N_k} \|\mathbf{a}(k)\|_2 + \epsilon \sum_{k=0}^{N_k} \sum_{s=0}^{N_o} c(k, s), \tag{29}$$

subject to:

$$\mathbf{r}(0) = \mathbf{r}_0 \tag{30}$$

$$\mathbf{v}(0) = \mathbf{v}_0 \tag{31}$$

$$\mathbf{v}(k+1) = \mathbf{v}(k) + \mathbf{a}(k)\delta t \tag{32}$$

$$\mathbf{r}(k+1) = \mathbf{r}(k) + \mathbf{v}(k)\delta t + \frac{1}{2}\mathbf{a}(k)\delta t^2 \tag{33}$$

$$\|\mathbf{a}(k)\|_2 \leq a_{max} \tag{34}$$

$$\|\mathbf{v}(k)\|_2 \leq v_{max} \tag{35}$$

$$\|\mathbf{r}(N_k) - \mathbf{r}_{target}\|_\infty \leq D_T \tag{36}$$

$$\|\mathbf{r}(k) - \mathbf{r}_{obs}(s)\|_2 \geq R_0 - c(k, s) \tag{37}$$

$$c(k, s) \geq 0$$

$$\forall\, k \in \{0, \ldots, (N_k - 1)\} \ , \ \ s \in \{1, N_o\} \tag{38}$$

where $\mathbf{r}(k)$ is the UAV's location at time-step k; $\mathbf{v}(k)$ is the UAV's velocity at time-step k; δt is the size of each time-step; $\mathbf{r}_{target}$ is the target location; $\mathbf{r}_{obs}(s)$ is the location of obstacle s; D_T is the distance at which the UAV is deemed to have reached the target; R_o is the radius of the obstacles. It should be noted that all positions are in a global frame of reference.

The cost function (29) primarily minimises time to target, N_k (in steps of k). There is also a small weighting, γ, on acceleration magnitudes to ensure a unique solution and a heavier weighting, ϵ, on the intrusion into each obstacle at each time-step, $c(k, s)$, representing the risk of detection and UAV loss. Obviously the value of the weights, γ and ϵ, will influence the observed behaviour of the controller. Setting $\gamma = 0.001$ and $\epsilon = 0.2$ was found to ensure that the obstacle region was never entered when only one obstacle was present while allowing for more direct routes to be taken when multiple obstacles overlap. It should be noted that the "behaviour" of the optimisation can be crudely controlled by adjusting the weights γ and ϵ and hence, the behaviour we are trying to reproduce or clone, however in the current context we are interested in the process of cloning a path planner and not on tuning what that behaviour is; the values selected represent an intuitive behaviour which is sufficient for the current application.

Equations (30) and (31) are initial condition constraints defining position, heading and speed. Equations (32) and (33) are a forward Euler approximation to the vehicle dynamics and kinematics respectively. Equations (34) and (35) are simple constraints on maximum velocity and acceleration. Equation (36) ensures that the UAV is within a given tolerance of the target at time N while (37) maintains a distance, $\|\mathbf{r}(k) - \mathbf{r}_{obs}(s)\|_2$, greater than or equal to $c(k, s)$ between the UAV and the obstacle s at time-step k. The $c(k, s)$ term can be used to selectively relax one or more of the intrusion constraints and guarantees stability and permits trajectories that pass through overlapping regions if the increased risk outweighs the additional time required to navigate the overlapping obstacles, i.e. the obstacles are soft constraints albeit with a high penalty. Finally, Equation (38) ensures that the optimisation does not 'reward' any points on a trajectory that are outside of an obstacle region.

Equation (37) is a non-convex constraint, and it is this feature that makes the problem so difficult. The two-norm constraint of (37) is implemented by approximating it with one of a choice of linear constraints, and using binary variables to selectively relax all but one of these as proposed in [28] and [5]:

$$\begin{aligned}
&\forall\, s \in \{1, \ldots, N_o\}, \quad \forall i \in \{1, \ldots, N_c\}, \\
&\forall\, k \in \{1, \ldots, N_k\} \\
&\mathbf{d_i^T}\,(\mathbf{r}(k) - \mathbf{r}_n(k)) \geq R_o - c(k, s) - M\,\mathbf{b}(s, i, k) \qquad (39) \\
&\forall\, s \in \{1, \ldots, N_o\}, \quad \forall k \in \{1, \ldots, N_k\} \\
&\sum_{i=1}^{N_c} \mathbf{b}(s, i, k) \leq N_c - 1 \qquad (40)
\end{aligned}$$

where N_c is the number of constraints that the circle is approximated by; M is a positive number that is much larger than any other term in the constraint; $\mathbf{b}$ is a set of binary decision variables (0 or 1); and:

$$\mathbf{d_i} = \begin{bmatrix} \cos\frac{2\pi i}{N_c} \\ \sin\frac{2\pi i}{N_c} \end{bmatrix} \qquad (41)$$

Equation (39) represents the linear approximations to Equation (37) and is equivalent to a N_c sided polygon located at the centre of the obstacle as illustrated in Fig. 5. Increasing N_c improves the accuracy of the approximation but at additional computational expense and it should be noted that despite the approximation, the obstacle is never violated. $N_c = 6$ was found to provide good compromise between accuracy and performance and is the value used in the remainder of this paper. Equation (40) ensures that at least one of the constraints is active at every time-step, i.e. the UAV must always remain outside the obstacle region in at least one direction, and becomes an extra constraint on the obstacle avoidance problem.

The resulting optimisation is a Mixed Integer Linear Programme (MILP) and is solved using the CPLEX optimisation package [65] via a Matlab interface. Repeated on-line re-planning at each time-step accounts for any model uncertainties and is a model predictive controller.

The algorithm below shows how the optimisation is implemented within MPC to achieve a feedback guidance law.

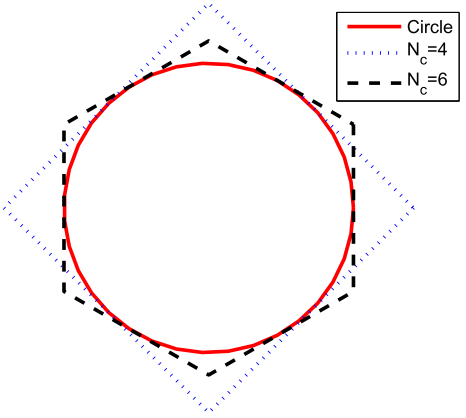


Fig. 5. Approximation of a circular obstacle with N_c linear constraints.

1. Convert $\{D_{target}, \theta_{target}, \boldsymbol{D}_{obstacles}, \theta_{obstacles}\}$ to $\mathbf{r}_{target}$ and $\mathbf{r}_{obstacles}$:

$$\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} D\sin\theta \\ D\cos\theta \end{bmatrix} \tag{42}$$

2. Solve optimisation (29) subject to (30)–(38)
3. Derive initial heading deviation, $\Delta\Psi_{MPC}$, from optimisation output:

$$\Psi_d = \tan^{-1}\left(\frac{v_y(k+1)}{v_x(k+1)}\right) - \frac{\pi}{2} \tag{43}$$

$$\Delta\Psi_{MPC} = \Psi_d - \Psi \tag{44}$$

4. Run simulation for δt, i.e. one time-step of the optimisation
5. Go to 1

In this work, the MILP optimizer is assumed to have full knowledge of the obstacles location, size and shape; in the case of multiple obstacles (Section 5), the optimizer is limited to information regarding only the closest two obstacles. In practice this is limited by the available sensors, however, the MPC–MILP approach can be made robust to these limitations by the addition of some simple terminal constraints [66]. For simplicity of the MILP, this extension has not been used in this paper. Similarly, moving obstacles have previously been addressed in MILP [67] and could be cloned in the same manner presented here, given appropriate attributes. The cloning process makes no assumptions on the nature of the optimizer, so the approach proposed can accommodate these and other extensions without modification.

4. Application of LDTs to UAV path planning

The standard LID3 algorithm [44] (summarised in Section 2.2.2) is based on assumptions about the nature of the data that do not hold given the current problem framework. This section presents novel extensions to the fundamental LID3 algorithm to account for circular attribute domains and to predict discontinuous output functions that are inherent to the UAV path planning problem.

4.1. Handling circular domains

This section identifies a limitation of linearly partitioning bearing type attributes and proposes an alternative method that is better able to capture the essential characteristic of a circular domain, i.e. it is modulo 360.

As previously discussed, distances and bearings were used to relate target and obstacle locations to the UAV's position, i.e. $\{D_{target}, \theta_{target}, \boldsymbol{D}_{obstacles}, \theta_{obstacles}\}$. The difficulty with a range/bearing representation is that bearings are defined in a polar coordinate system whilst the LID3 algorithm assumes a linear domain, for example the full fuzzy partitioning of a bearing attribute is illustrated in Fig. 6 with six fuzzy sets where the two terminal sets are highlighted.

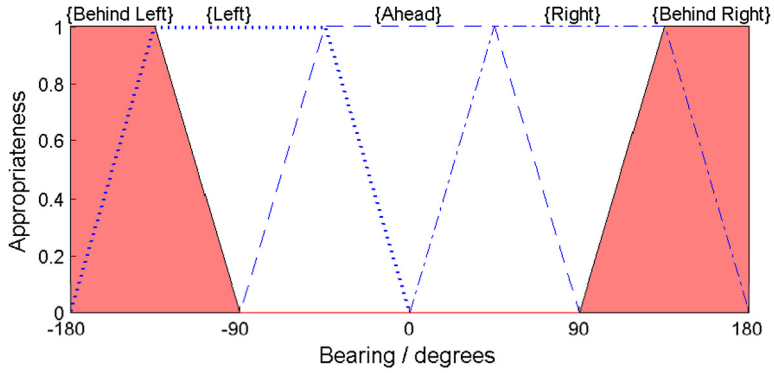


Fig. 6. Appropriateness measures for labels on a circular domain.

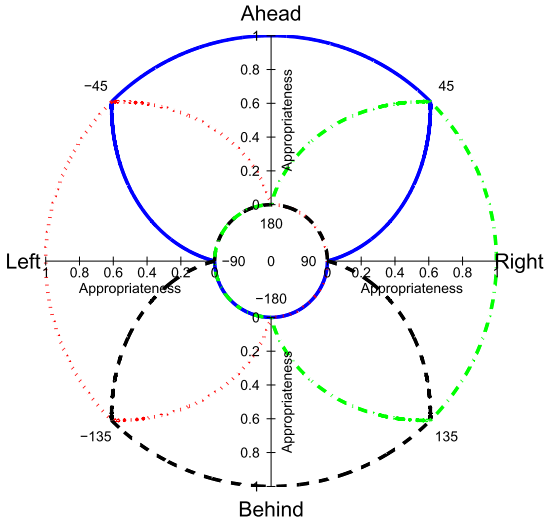


Fig. 7. Appropriateness measures for labels on a circular domain.

The representation shown in Fig. 6 satisfies the necessary conditions of the LID3 algorithm but fails to capture the discontinuity at the boundaries of the domain, i.e. $-180° = 180°$. Consequently, as the UAV changes heading the bearing to an obstacle (or target) can jump between the two extremes which is likely to lead to an inconsistent rule base as training data that represents very similar scenarios will result in very different forecasts.

The characteristic of bearing data that needs to be preserved is that it is circular, i.e. $-180°$ is equivalent to $180°$ and vice versa. This can be achieved by merging the two boundary fuzzy sets from a linear domain such as that in Fig. 6, to give the coverage shown in the polar plot in Fig. 7 where angle denotes bearing and radius denotes the appropriateness of each label; the equivalent mapping fuzzy partitioning on the linear domain is the equivalent of that shown in Fig. 6 but considering the two terminal sets, {Behind Left} and {Behind Right}, as a single set {Behind}, giving the same labels as in the polar case (Fig. 7).

Fig. 7 illustrates the proposed approach to ensuring a full fuzzy partitioning of a circular domain using only four labels. In order to maximise performance, the cloned LDT controller in this paper uses ten focal elements on each bearing type attribute. The membership function of ten focal elements covering 360° is shown in Fig. 8 with three focal elements highlighted, note: the transition from fuzzy labels in Fig. 7 to focal elements in Fig. 8; the maximum membership for any angle is in the centre of the diagram; the sum of the memberships for any angle equals one. The angle covered by the focal elements (domain of discourse) represents the angle from the UAV flightpath angle of an obstacle or target with 0° representing an object on the current flightpath. The linguistic interpretation of each focal element is best expressed in terms "clock position" [68] (also indicated on the figure) such that an object directly behind the UAV could be described as being at "6 o'clock". Due to the number of focal elements selected to maximise

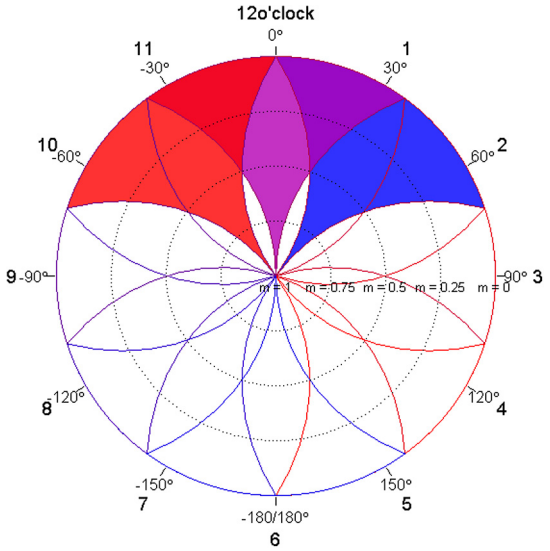


Fig. 8. Focal element membership functions for cloned LDT controller.

performance, there is not one representing 9 and 3 o'clock, this may reduce interpretability slightly but could be adjusted if this was considered more important than performance.

4.2. Discontinuities in output: modal-LID3

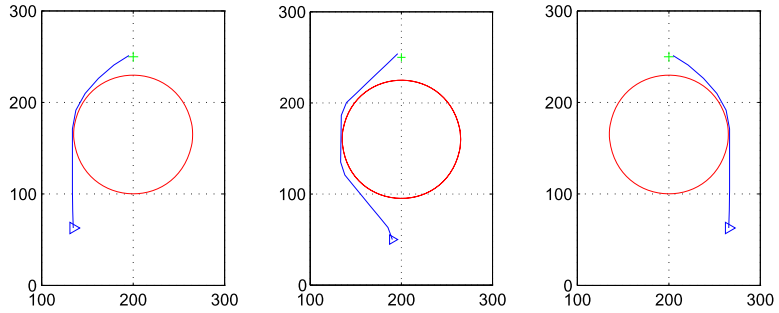
The single obstacle avoidance problem exhibits a discontinuity in the required heading deviation either side of the target-threat centre line where the shortest path switches from left to right of the threat (illustrated in Fig. 9). As the discontinuity is approached the probability distribution over the output focal elements becomes bi-modal (Fig. 9(c)) reflecting the cost function of the optimisation where the difference in cost between heading left or right of the obstacle tends to zero. The introduction of additional obstacles leads to further discontinuities as discussed in Section 4.3.

Close to the discontinuity, nearby locations may have quite different resolutions (left or right), hence the decision tree indicates roughly equal likelihood of either direction as shown in the central graph of Fig. 9(c). The LID3 defuzzification procedure (Eqn. (24)) assumes a linear output function so effectively interpolates between the two maxima resulting in approximately zero heading deviation. Consequently the LDT prediction is poor and the UAV flies almost straight through the centre of the obstacle. Obviously such behaviour is unacceptable and the standard LID3 algorithm must be modified.

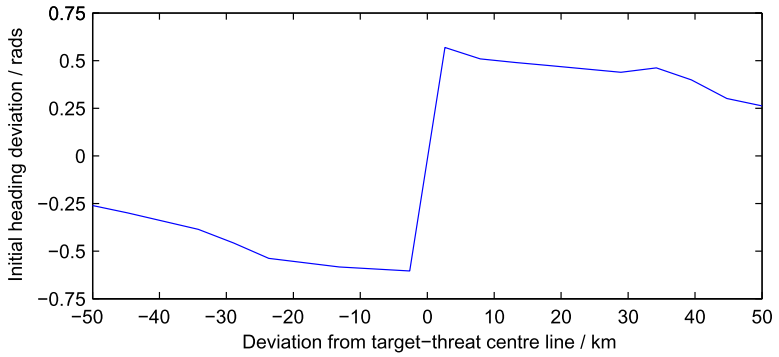
To improve resolution between conflicting decisions, we consider only the part of the target attribute probability distribution that "belongs to the most probable mode". The region of interest can be determined by finding the most probable focal element on the target attribute and selecting the focal elements either side of the maximum that have monotonically decreasing probability. For a distribution with a single mode this method will be equivalent to the original LID3 defuzzification process (24).

The expected value of the selected region of the distribution is defined in (45), where: $\hat{y}$ is the estimated output value; F_t^n and F_t^m are the focal elements at either end of the section of the probability distribution we are interested in; $P(F_t^i|\mathbf{x})$ is the probability of target focal element F_t^i given the current input; and $E(y|F_t^i)$ is the expected value of y given focal element F_t^i.

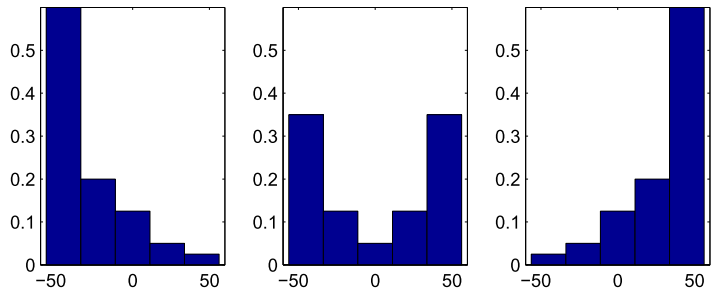
$$\hat{y} = \frac{\sum_{i=|F_t^m|}^{|F_t^n|} P(F_t^i|\mathbf{x}) E(y|F_t^i)}{\sum_{j=|F_t^m|}^{|F_t^n|} P(F_t^j|\mathbf{x})} \tag{45}$$



(a) Example scenarios



(b) Output of optimisation around the discontinuity



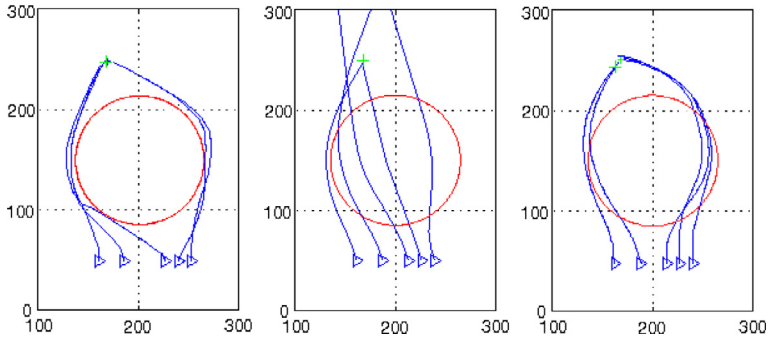
(c) Example probability distributions on target attribute focal elements

Fig. 9. Illustration of discontinuity at target-threat centre line.

The denominator is a normalisation constant required to ensure the sum of the probabilities of the selected focal elements is unity. LID3 based on this alternative defuzzification will subsequently be referred to as Modal-LID3.

Observing the trajectories generated around the discontinuity by the original and modal-LID3 algorithms gives a general impression of their performance (Fig. 10). From the trajectories it is clear that Modal-LID3 makes a much closer approximation to the optimised trajectories than the un-modified algorithm.

A clearer indication of the controller's performance can be obtained by plotting the predicted heading deviation at the first time-step against the distance from the target-threat centre line (Fig. 11). This shows that the Modal-LID3 controller makes a very good approximation to the discontinuity and that the performance is maintained as the UAV moves away from the target-obstacle centre line. The smaller step changes in the predicted heading deviation occur when the probability distribution switches from bi-modal to uni-modal and all of the target focal elements start to contribute to the tree output.



(a) Optimisation (b) LID3 (c) Modal-LID3

Fig. 10. Illustration of decision tree performance compared to optimisation as discontinuity is approached.

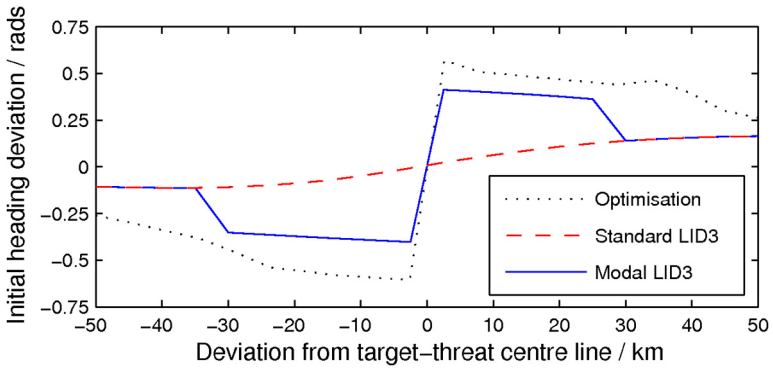


Fig. 11. Comparison of defuzzification methods around a discontinuity.

4.3. Multiple discontinuities

The introduction of additional obstacles increases both the number and complexity of the discontinuity decision. The number of discontinuities increases as the number of disconnected classes [18] increases from two in the single obstacle case, *left* or *right* of the obstacle, to three in the two obstacle case, *left*, *right* or *between* (Fig. 12). Clearly this complexity would increase exponentially with the number of obstacles.

The complexity of the discontinuity decision also increases as the shortest path to the target can be dependent on both obstacle locations. Consider Fig. 12(c): if there was only the obstacle closest to the UAV's initial position then the shortest path to the target would pass to the right of this obstacle; the addition of the second obstacle causes the optimal trajectory to switch to the left.

In order to assess if the modified LID3 algorithm is able to handle multiple as well as single discontinuities, the performance of the decision tree controllers was evaluated over a test set of scenarios containing two obstacles. The scenarios were generated with a fixed initial position, target location, and one obstacle location; the second obstacle location was varied from 150 km to the left, to 150 km to the right of the first obstacle and a fixed distance in front of the UAV's initial position. The initial heading of the UAV was set to due North in all cases. Fig. 12 shows a selection of the generated scenarios with optimised paths generated by the MPC based controller.

The optimum LDT parameters were determined by varying each one in turn over a range of values as in previous work [33,69]: range attributes were covered by 17 focal elements; bearing attributes by 10; and the target attribute (heading deviation) by 7. The tree depth was set to the maximum of 6 as empirical results suggested this would yield the best performance for these scenarios.

Section 4.1 presented an interpretation of bearing attributes. A similar interpretation of range bearings can be made. Fig. 14 shows the "range to the target" as an example of a range attribute: the attribute domain spans from approximately 6 km to 350 km and is covered by 17 focal elements as stated previously. The labels for each focal element are presented along the top of each figure and are equivalent to the approximate time that would be required

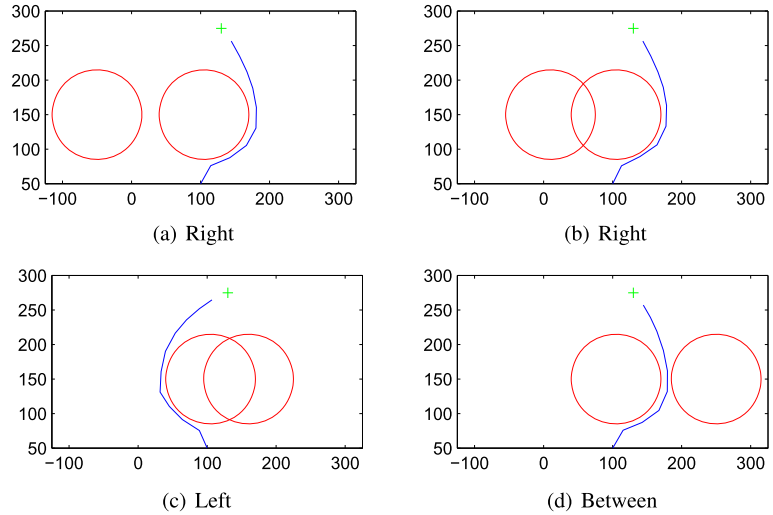


Fig. 12. Illustration of extra discontinuity in initial heading deviation and the notion of disconnected classes of paths.

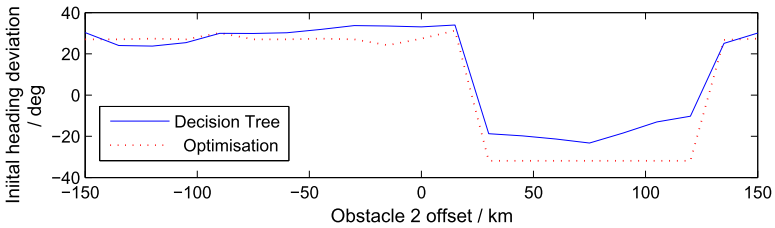


Fig. 13. Decision tree approximation to discontinuities that occur as the 2nd obstacle location is varied from the left to the right of the UAV's initial position.

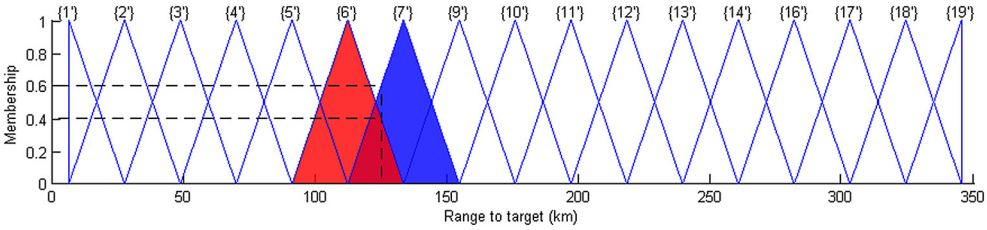


Fig. 14. Focal elements covering the "range to target" attribute domain.

by the UAV to travel that distance, this is able to convey the proximity of an object whilst maintaining a degree of imprecision, for example the focal elements that cover a distance of 125 km are highlighted and can be interpreted to mean that "the target is between approximately 6 and 7 minutes away" (with membership of 0.4 and 0.6 respectively). It should be noted that the labels given to the elements represent the most appropriate time to the target to the nearest minute; given the size of the domain, speed of the vehicle and number of focal elements this does not form a continuous sequence (note the absence of 8 and 15) which arguably reduces interpretability. Should this be considered a problem then increasing the number of focal elements would be a simple solution.

Fig. 15 shows the focal elements that cover the target attribute, "heading deviation" which requires far fewer focal elements such that the labels are a simpler linguistic description. It is noted that triangular focal elements are used for all attributes, this helps to improve interpretability compared to more complex shapes such as Gaussian. Furthermore, the LDT algorithm is less sensitive to the shape of the focal element (cf. fuzzy set) than some algorithms are as the output is a weighted probability distribution so defuzzification produces a piecewise linear function.

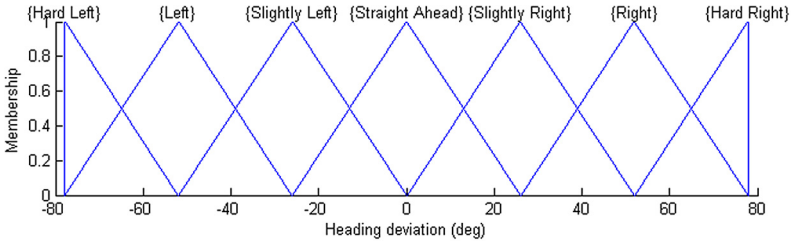


Fig. 15. Focal elements covering the "heading deviation" attribute domain.

Table 1
Mean and standard deviation of r.m.s. errors of selected algorithms for ten-old cross validation.

Algorithm	RMS error/degrees $\bar{x} \pm \sigma$
LDT	8.60 ± 1.43
NN	7.84 ± 1.60
SVM	15.65 ± 1.06
RT	8.81 ± 1.78

The performance of the MPC and LDT controllers are compared in Fig. 13 for the range of scenarios described above. The initial heading deviation of the optimisation clearly shows the two discontinuities where the path switches from right to left and back to the right of the fixed obstacle. The LDT successfully reproduces the optimisation behaviour for all locations of the second obstacle although the magnitude of the turn when the trajectory passes to the left of the obstacles is slightly smaller but this may be accounted for at the second time-step.

Having demonstrated the ability of the LDT to capture multiple discontinuities its performance is now compared with alternative methodologies.

4.4. Validation of LDT performance

In order to gain confidence in the performance of the LDT algorithm, 10-fold cross-validation was performed on a training set with 1000 examples taken from scenarios with a single obstacle, i.e. each example consists of four attributes and one target attribute. The tree parameters used were 12 focal elements on all input attributes, 5 focal elements on the output attribute (heading deviation) and the tree was only generated to depth 3 largely due to the training data only consisting of a single threat.

Table 1 presents the r.m.s. error of the LDT on each of the 10 test sets along with the error using the same training and test data for a Neural Network (NN), Support Vector Machine (SVM) and Regression Trees (RT). The NN was implemented using the Neural Network toolbox in MATLAB® and contained 10 hidden nodes. The SVM was implemented using the Matlab® interface to the LIBSVM library [70] to run an epsilon-SVR formulation with a radial basis function. The regression tree was implemented using Matlab® Statistics toolbox implementation of [71]. All other parameters for the three methods were left at their default settings.

The results in Table 1 suggest that the performance of the LDT is comparable to the NN and RT but better than the SVM. To confirm this observation a two sample t-test at the 95% significance level was performed on the errors produced by each method with the null hypothesis that the errors are from the same distribution. Results of the t-tests are presented in Table 2 and confirm the conclusions made previously. Furthermore, it is apparent that the SVM is outperformed by all other methods and also that the neural network outperforms all methods *except* for the LDT.

The ability of the LDT to match or exceed the performance of the (un-tuned) black-box methods, which are designed to maximise performance but have very low interpretability, shows that the performance of the LDT is at least comparable to the more established methods.

From the results it is clear that the LDT is able to make as good an approximation to the test data as alternative methods but is not significantly "better". It is noted that the method proposed in this paper is intended as an example of behavioural cloning in a context where an interpretable controller is required and it is accepted that, with more

Table 2
Comparison of alternative algorithms using a two sample t-test at the 95% confidence level where acceptance of the null hypothesis (that the performance of the algorithms is the same) is indicated with ✓ and rejection by ✗.

	LDT	NN	SVM	RT
LDT	–	✓	✗	✓
NN	✓	–	✗	✗
SVM	✗	✗	–	✗
RT	✓	✗	✗	–

thorough tuning, other methods may be able to offer improved performance at the expense of interpretability. However, it is also noted that later sections demonstrate the ability of the decision tree to generalise well beyond the training data especially when the prediction algorithm is modified in cases with high uncertainty.

This section has demonstrated that the LDT controller is able to accurately reproduce complex output functions including multiple discontinuities. It is well established that extending the tree to near the maximum depth can result in over-fitting the training data leading to poor generalisation [72]. Consequently we now consider the controller's performance on a more general set of test scenarios to demonstrate robustness against the training parameters and process.

5. Generalisation to N obstacles

Having demonstrated that the decision tree is capable of capturing the discontinuity associated with the second obstacle, this section considers the controller's performance on the general n obstacle problem. The complexity of generating the decision tree increases exponentially with the number of attributes and, for a given training database, the data upon which the probability distribution of each branch is calculated decreases exponentially with depth. Consequently it is desirable to limit the number of attributes and depth of the decision tree controller.

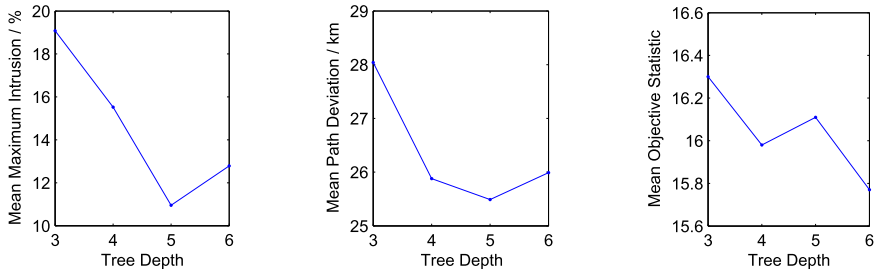
To avoid this growth in complexity, both in training and online decisions, we approximate the full n-obstacle controller using the 2-obstacle controller, fed with data on only the two nearest obstacles. Initial investigations using the full MPC–MILP law suggest that this assumption is reasonable, when tested using a Wilcoxon rank-sum test at the 5% significance level to compare the trajectories generated on complete and reduced information. To implement this approximation a data selection module needs to be added to the controller to identify the two nearest obstacles and then supply the range and bearing of those obstacles as inputs to the decision tree controller.

Section 5.1 identifies a set of tree parameters that results in good performance on the general 2-obstacle problem. Section 5.2 then investigates the ability of this controller to generalise to n obstacles.

5.1. Tree depth

In the previous section it was shown that the discontinuities for the two obstacle case can be accurately reproduced with a fully expanded tree. However, in order to improve generalisation it is often desirable to limit the depth of the tree. To test if this is the case in the two obstacle case, trees of depth 3 to 6 were used to generate trajectories for 100 test scenarios containing two obstacles; the training data for all the trees was identical and consisted of nearly 2500 examples taken from 200 different scenarios involving 2 obstacles. The results of the comparison are presented in Fig. 16 using three statistics: the *maximum intrusion* is the maximum intrusion of a trajectory in a given scenario; the *path deviation* is the mean distance at each time-step between the optimised and decision tree trajectories for a given scenario; the *objective statistic* is a weighted combination of path length and total obstacle intrusion at each time-step for a given scenario.

The results clearly show that depth 3 controller performs least well according to all the statistics. The effect of over-fitting can be seen most clearly in the depth 6 controller results where the path deviation and maximum intrusion statistics show a noticeable deterioration in performance compared to depth 5. The relative performance of the depth 4–6 controllers varies with each statistic. Given that there is no obvious choice, depth 4 was selected to be used in



(a) Mean maximum intrusion (b) Mean path deviation (c) Mean value of the objective statistic

Fig. 16. Effect of decision tree depth on controller performance on a set of 100 test scenarios.

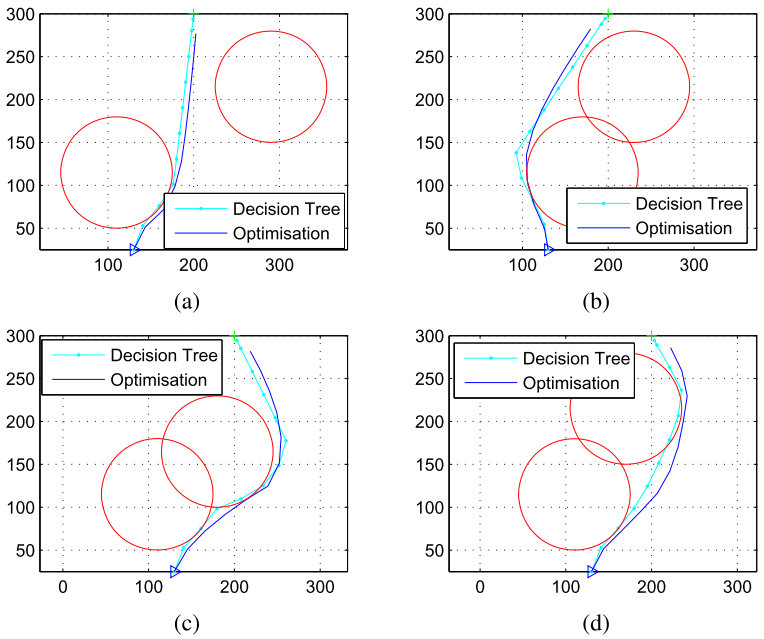


Fig. 17. Some examples of generalisation to two obstacles.

the remainder of the experiments on the basis that it should demonstrate the best generalisation whilst improving transparency and computational efficiency.

Fig. 17 shows some typical results from a depth 4 decision tree (hashed line) and the optimised paths (solid line) for the same scenarios. In some of the examples such as Figs. 17(a) and 17(b) the trajectory is only dependent on one obstacle, i.e. the 2nd obstacle places no constraints on the UAV trajectory. In other cases such as Figs. 17(c) and 17(d) the 2nd obstacle constrains the optimised trajectory causing it to pass, at least one of the obstacles, on the opposite side compared to a scenario without it. This is further evidence that the decision tree controller is capable of capturing the more complex two obstacle behaviour of the optimal controller.

5.2. *Effect of number of obstacles on performance*

To test the controller's ability to generalise to multiple obstacles, 5 sets of 100 test scenarios were generated. Each set contained scenarios consisting of 3–7 obstacles. Trajectories for all of the test scenarios were generated by both the decision tree and the optimised controllers, using the assumption that the globally optimal trajectory can be approximated by the nearest two obstacles.

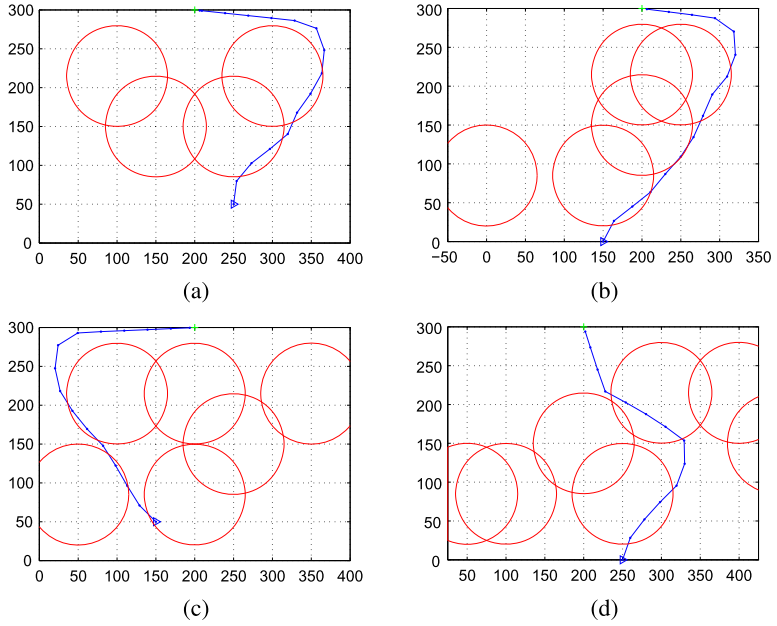


Fig. 18. Some examples of LDT generalisation to multiple obstacles.

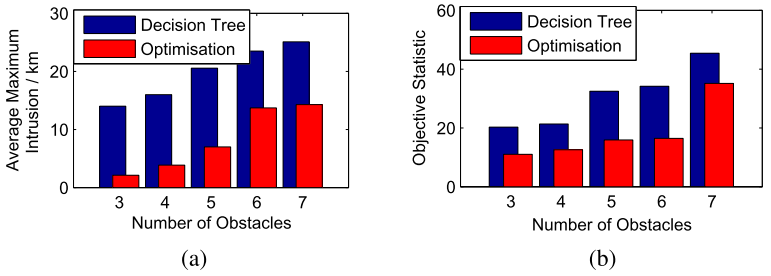


Fig. 19. Comparison of decision tree and optimised controllers on scenarios with multiple obstacles.

Fig. 18 shows some examples where the decision tree controller generates good trajectories for scenarios with 4–7 obstacles, either skirting the periphery of all the obstacles or exploiting a region of low risk to reduce the trajectory length. The scenario shown in Fig. 18(d) is a good example of where overlapping obstacles require that the UAV's trajectory passes through an obstacle region in order to reach the target.

To compare the decision tree performance against that of the optimal controller, the maximum intrusion and objective statistics were calculated for both controllers on each of the test sets, the results are shown below in Fig. 19. Both graphs show that the relative performance of the decision tree and optimal controllers remain approximately constant which shows that the decision tree is able to generalise to a higher number of obstacles. However, the degree to which the optimisation out performs the decision tree is large, greater than 50% in some instances, and cannot solely be justified by the tree being only an approximation to the optimised behaviour.

Section 6 uses the interpretability of the decision tree to understand what factors are leading to the reduced performance of the decision tree when generalising to a large number of obstacles as well as presenting some interpretability measures for the LDT controller.

6. Interpretability

Fuzzy systems can be considered inherently transparent [9] as their behaviour can be explained in terms of their components and their relations. However, the focus on improving accuracy [11,73] through techniques such as neuro-

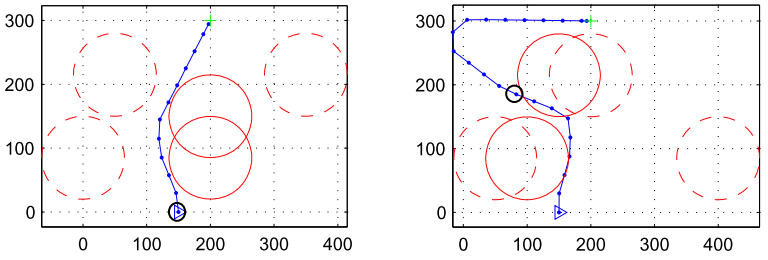


Fig. 20. Typical decision tree trajectories for 5 obstacles.

Table 3
Active rules for initial decision of the trajectory in Fig. 20(a).

$P(B_1\|\mathbf{x}) = 0.33$	$P(B_2\|\mathbf{x}) = 0.23$
Obs_1 is about 5 mins 45 secs distant AND Obs_2 is at 12 o'clock AND the *target* is at 12 o'clock AND Obs_1 is at 1 o'clock	Obs_1 is about 5 mins 45 secs distant AND Obs_2 is at 12 o'clock AND the *target* is at 12 o'clock AND Obs_1 is at 12 o'clock
{(hard left, 0), (left, 0.03), (slightly left, 0.41), (straight ahead, 0.30), (slightly right, 0.18), (right, 0.08), (hard right, 0)}	{(hard left, 0), (left, 0.05), (slightly left, 0.23), (straight ahead, 0.06), (slightly right, 0.58), (right, 0.08), (hard right, 0)}
$P(B_3\|\mathbf{x}) = 0.11$	$P(B_4\|\mathbf{x}) = 0.11$
Obs_1 is about 5 mins away AND the *target* is at 12 o'clock AND Obs_2 is at 12 o'clock AND Obs_1 is at 1 o'clock	Obs_1 is about 5 mins 45 secs distant AND Obs_2 is at 1 o'clock AND the *target* is at 12 o'clock AND Obs_1 is at 1 o'clock
{(hard left, 0), (left, 0.03), (slightly left, 0.29), (straight ahead, 0.35), (slightly right, 0.12), (right, 0.19), (hard right, 0.01)}	{(hard left, 0), (left, 0.02), (slightly left, 0.33), (straight ahead, 0.62), (slightly right, 0.02), (right, 0.002), (hard right, 0)}
SO TURN slightly left (0.32) AND straight ahead (0.25) $\Rightarrow \Delta\Psi = -17.3^\circ$	

fuzzy [74] tended towards models that were unintelligible, i.e. they could not be understood by a user and lacked interpretability.

Given that there is not even an agreed definition of interpretability [9] measuring it is even harder [75]. This section demonstrates, through the use of examples, that the decision tree controller offers a high degree of interpretability which enables the reasoning behind the controller's decision making to be understood before providing some interpretability measures [76] typical of the LDTs used in this paper.

6.1. Demonstration of LDT interpretation

In order to demonstrate the interpretability of a LDT, several examples are provided and discussed. Fig. 20 shows some example trajectories generated by the LDT controller where the obstacles shown with a solid line are the nearest two at the circled time-step.

Fig. 20(a) shows an example where the decision tree generates a good trajectory that takes the shortest route to the target and which avoids all the obstacles. Examining the rules that the initial decision is based on provides a clear insight into the algorithms reasoning. Table 3 shows the four branches that best describe the current UAV state (for the circled position in Fig. 20(a)). In addition to the linguistic description of the branch, the probability distribution

Table 4
Example of active rules in an area of high uncertainty.

$P(B_1\|x) = 0.30$	$P(B_2\|x) = 0.18$
Obs_1 is about 4 mins 15 secs distant AND the *target* is at 4 o'clock AND Obs_1 is at 5 o'clock AND Obs_2 is at 8 o'clock	Obs_1 is about 4 mins 15 secs distant AND the *target* is at 4 o'clock AND Obs_1 is at 5 o'clock AND Obs_2 is at 7 o'clock
{(hard left, 0.14), (left, 0.14), (slightly left, 0.14), (straight ahead, 0.14), (slightly right, 0.14), (right, 0.14), (hard right, 0.14)}	{(hard left, 0.14), (left, 0.14), (slightly left, 0.14), (straight ahead, 0.14), (slightly right, 0.14), (right, 0.14), (hard right, 0.14)}
$P(B_3\|x) = 0.15$	$P(B_4\|x) = 0.08$
Obs_1 is about 4 mins 15 secs distant AND the *target* is at 4 o'clock AND Obs_1 is at 4 o'clock AND Obs_2 is about 5 mins 30 secs	Obs_1 is about 3 mins 30 secs distant AND obs_1 is at 5 o'clock AND the *target* is at 4 o'clock AND Obs_2 is at 8 o'clock
{(hard left, 0), (left, 0), (slightly left, 0.001), (straight ahead, 0.22), (slightly right, 0.75), (right, 0.025), (hard right, 0)}	{(hard left, 0.14), (left, 0.14), (slightly left, 0.14), (straight ahead, 0.14), (slightly right, 0.14), (right, 0.14), (hard right, 0.14)}
SO head slightly right (0.22) AND straight on (0.16) $\Rightarrow \Delta\Psi = 3.9°$	

on the output variable given each branch is also presented. The output attribute probability distribution is aggregated over the entire tree and defuzzified according to (45). The final line of the table presents a linguistic interpretation of the two most likely actions according to (45) and the resulting heading deviation.

The order of the clauses of each branch in Table 3 reflects the ordering of the attributes in the tree. In order to improve the interpretability it may be simpler to compound some of the statements, for example, the rule for B_1 could be written as "Obs_1 is at 1 o'clock and about 5 mins 45 secs distant AND Obs_2 AND the *target* are both at 12 o'clock".

Fig. 20(b) shows a scenario where the decision tree clearly generates a sub-optimal trajectory. The initial poor decision occurs 4 time-steps before the circled one and is due to the limited awareness of the controller, i.e. only aware of the nearest two obstacles. However, once the UAV has moved away from the optimised trajectory it enters a region of the problem space for which there is very limited training data, i.e. high uncertainty. This can be seen by examining the probability distribution on the output attribute given each branch and is shown below for the circled time-step where the probability distribution is uniform in all cases except one. This is indicative of sparse training data and is a common cause of sub-optimal trajectories generated by the decision tree (see Table 4).

6.2. Interpretability measures

Many taxonomies to compare the interpretability of different algorithms have been proposed [9,10,12,76]. This paper adopts the "quadrant" approach of [76] which aims to distinguish between the, often conflicting, complexity-based and semantic-based interpretability measures at the rule base and fuzzy partition levels such that any given measure belongs to one of the following quadrants:

- Q1: Complexity-based measures at the rule base level;
- Q2: Complexity-based measures at the fuzzy partition level;

- Q3: Semantic-based measures at the rule base level;
- Q4: Semantic-based measures at the fuzzy partition level.

Typical Q1 measures are the *number of rules* and the *number of conditions* in the antecedent. The number of rules in the rule base of a LDT is the total number of branches, i.e. a product of the number of focal elements on each attribute and the tree depth. In the current application, the trees were generated to a maximum depth of four with a maximum of 17 focal elements on the attributes. Consequently the maximum number of rules would be $17^4 = 83\,521$, however this value will never be attained as some branches will not be expanded to the maximum depth and not all attributes have 17 focal elements. While the number of rules is high, the maximum number of conditions that are contained within each rule, i.e. the tree depth of four, is very low.

Common interpretability measures in Q2 are the *number of features* and the *number of membership functions*. The proposed model makes predictions based on only the range and bearing to the target and two nearest obstacles, i.e. there are six features in the model which is within the recognized 7 ± 2 distinct conceptual entities most humans can handle [77]. The same is not true when considering the number of membership functions, i.e. focal elements, where different attributes are partitioned into 17 or 10 focal elements. However, Section 6.1 demonstrated a framework commonly used in a military context that the focal elements readily translate to, i.e. directions based on a clock and distances measured in time. The interpretability of the LDT used in this paper could be further improved by improving this correlation even further, i.e. 19 and 12 focal elements (corresponding to distances of up to 19 minutes and directions aligned with each "hour" on a clock-face).

Semantic-based interpretability at the rule base level, i.e. Q3, can be measured in terms of the *number of rules fired simultaneously* or the *consistency of the rule base*. LDTs can fire a maximum of 2^n rules simultaneously, where n is the depth of the tree, i.e. 16 rules in the current context. Whilst 16 rules could be considered manageable, it has been illustrated in Section 6.1 that typically only approximately four rules contribute significantly to the predicted output resulting in a very high degree of interpretability. The consistency of the rule base is hard to measure and it should be noted that the problem does not lend itself to such a measure due to the discontinuous nature of the output.

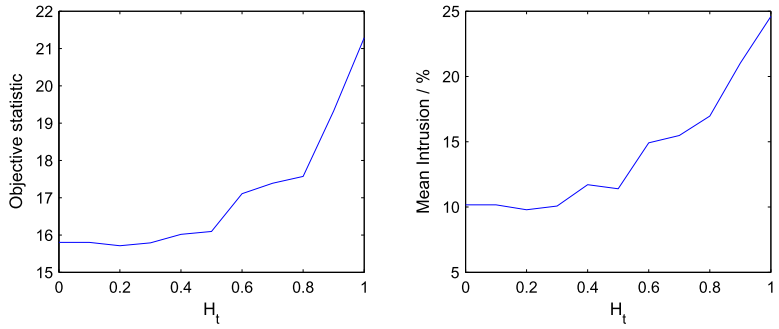
Q4 accounts for semantic-based interpretability at the fuzzy partition level and includes such measures such as *completeness* and *normalization*. All of the criteria in Q4 are met to the highest level where the model uses strong fuzzy partitions, as this is the case in LID3 then it is clear that the model demonstrates a high degree of interpretability as measured by these criteria.

Taking a global view of the interpretability measures it is clear that the LDT used in this paper has a high semantic-based interpretability (Q3 and Q4) while its complexity-based interpretability (Q1 and Q2) is more mixed, scoring well by some metrics and poorly by some others. This distinction between the semantic and complexity-based interpretabililty of the algorithm used in this context is perhaps to be expected: in order to achieve a good prediction of such complex function the model must itself have a degree of complexity, however, by maintaining clear semantic interpretability and structuring the model to reduce some aspects of complexity the desired combination of good performance and high interpretability has been achieved.

Section 6.1 identified that despite attempts to generate a training set that was representative of all scenarios, when generalising to multiple threats the LDT often encounters situations of high uncertainty, i.e. little to zero relevant training data. One approach to improve performance in these instances would be to generate more training examples, however, this assumes that the new training set will now be "complete". Section 7 proposes a modification to the LDT algorithm itself in order to improve the performance in regions of high uncertainty, thus reducing the dependence on the training data.

7. Forward planning horizon

Section 5 demonstrated that the LDT based controller can make good approximations to the optimised trajectories for the multi-obstacle case. However, Section 6 identified that the LDT performance can be limited by regions of high uncertainty. This section proposes an alternative method that attempts to improve performance in such situations before comparing the computational complexity of the new algorithm with the MPC–MILP reference controller.



(a) Objective Statistic (b) Percentage Intrusion

Fig. 21. Effect of entropy threshold on performance.

7.1. Lookahead-LID3

A decision tree makes a prediction based on the current state and past examples. If the number of reachable states can be reduced to a manageable set, then an alternative approach is to evaluate each state in turn and select the one that is deemed "best".

One method for selecting the "best" heading deviation from a set of possible actions is to identify the one with the lowest tree entropy. This equates to progressing towards regions of low uncertainty, i.e. regions with a high number of training examples. By definition, all data points lie on optimal paths, and hence minimising the UAV's distance from the training examples also minimises the distance from the optimised trajectory for the current state. If the data points were randomly located throughout the space, the lookahead algorithm is unlikely to offer a significant improvement. Such a controller is synonymous with a receding horizon optimisation to minimise the expected uncertainty, with a one step planning horizon.

The entropy of the output probability distribution provides a measure of the uncertainty in any given prediction. Consequently a simple threshold (H_t) can be used to specify the level of uncertainty (entropy) required in the output probability distribution before the lookahead algorithm is applied. When $H_t = 1.0$ the lookahead algorithm is never applied and is equivalent to the unmodified decision tree while $H_t = 0$ would cause the controller to lookahead at every time-step. By including $\Delta\Psi = \Delta\Psi_{predicted}$ in the set of actions that the lookahead algorithm evaluates, it is possible to reduce the probability of making a poorer decision than the original decision tree prediction. The proposed control scheme is as follows:

1. Calculate probability distribution on target attribute
2. Test probability distribution entropy (H)
   (a) $H < H_t$: make prediction based on entropy distribution
   (b) $H \geqslant H_t$: evaluate n evenly spaced heading deviations over $[-\Delta\Psi_{\max}, \Delta\Psi_{\max}]$ and also $\Delta\Psi = \Delta\Psi_{predicted}$, then select the one with the lowest entropy.

The optimum value for H_t was investigated by comparing the performance of different controllers on 100 test scenarios with two obstacles. Fig. 21 shows that the results of the comparison based on the objective function and intrusion statistics. An improvement in the controller performance is reflected by a lower value of either statistic and the graphs show that performance decreases exponentially between approximately $H_t = 0.2$ and 1.

Applying the lookahead algorithm would be expected to improve the controller performance as it evaluates several trajectories in addition to the one predicted by the tree. This is reflected by the general trend of the graphs in Fig. 21. However, very low tree uncertainty implies a high confidence that the tree prediction is optimal. In such cases, the lookahead algorithm is unlikely to be able to make a significant improvement on the tree prediction. This is reflected in the gradient of the graphs: a small increase in the entropy threshold from zero, i.e the lookahead algorithm is no longer applied in cases of low uncertainty, shows no degradation and possibly a slight improvement in performance up to $H_t = 0.2$; as the threshold tends to one, i.e. the lookahead algorithm is only applied in cases of high uncertainty, the performance deteriorates rapidly as the lookahead algorithm is utilised in ever fewer situations. It is also noted

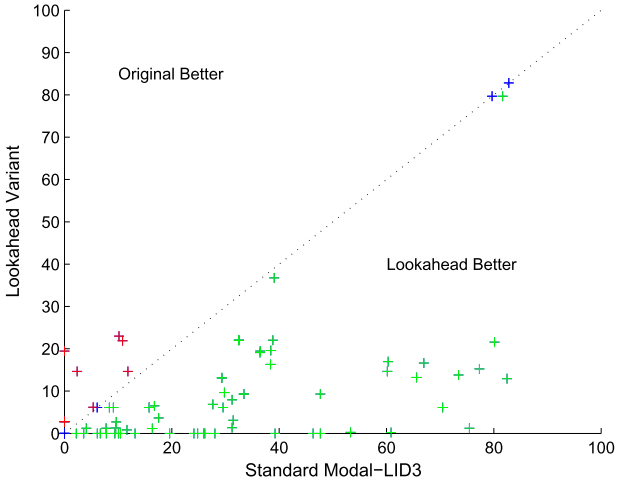


Fig. 22. Performance of modified controller where $H_t = 0.2$.

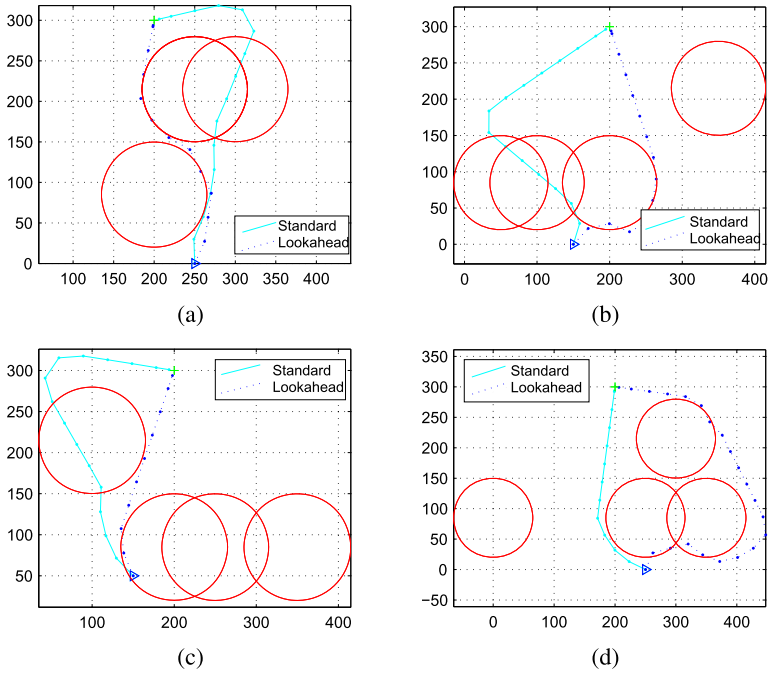


Fig. 23. Examples with an entropy threshold of 0.2.

that applying the lookahead algorithm adds additional complexity to the decision making process and hence is likely to reduce interpretability of the decision at the affected time-step(s).

Fig. 22 compares the performance of the standard and modified controller (with $H_t = 0.2$) for each of the 100 test scenarios. In 82% of cases the modified controller equals or out-performs the standard algorithm. Perhaps the most interesting trend is the number of cases where the intrusion statistic is reduced to zero, even where previously the algorithm may have intruded up to 60%.

Examining some of the scenarios from the modified controller gives an impression of the effect of the algorithm. Fig. 23 illustrates some scenarios where the modified controller with a threshold of 0.2 has altered the behaviour and shows that the potential improvement to a trajectory is very high while, even in the cases where the performance deteriorates the most, the deterioration is comparatively small and importantly the algorithm is able to recover from

Table 5
Comparison of optimisation and decision tree decision period.

	Decision period (ms)		
	Min	Mean	Max
Optimisation	171	597	19 593
Decision tree	<1	4	7

Table 6
Comparison of optimisation and decision tree objective statistic.

	Objective statistic		
	Min	Mean	Max
Optimisation	9	12.48	32.57
Decision tree	9	17.00	62.82

an initial bad decision. The ability to recover from a bad decision is the most notable difference from the standard algorithm where subsequent time-steps tend to exacerbate a poor decision.

7.2. Computational complexity

The motivation for cloning the MPC–MILP controller was to derive an equivalent controller that was additionally suitable for use in a real-time system, consequently it is necessary that the computational complexity of both algorithms are examined. Previous sections have discussed the performance of the LDT controller but this must also be considered when comparing the computational complexity so is briefly considered again at the end of this section.

The computational complexity of the MPC–MILP and Lookahead-LDT controllers was assessed as the time taken to make a prediction at each time-step. Both controllers were used to generate heading deviation commands as an input to the simulation introduced in Section 2.3 for 100 scenarios based on knowledge of the two nearest obstacles. Each scenario contained a total of five obstacles pseudo-randomly located over 25 locations (that ensured overlap between obstacles) and the nearest two were evaluated prior to making a decision at each time-step. The UAV initial position was also selected at random from 4 fixed points to ensure that at least a subset of the obstacles were located between the initial position and target location (which was fixed). All obstacles, initial positions and target location were contained within a 400×300 km region. All obstacles had a radius of 65 km.

The hardware used was a P4 3.2 GHz PC with 1 GB of RAM running Windows XP. The optimal controller was called and timed from within Matlab while the decision tree was timed within a standalone program. To make the comparison as fair as possible the optimisation was run outside of the Matlab environment although there is a small overhead associated with the setting up of a new process which will be included in the results observed.

Table 5 compares the computational complexity of the MPC–MILP and LDT based controllers. The mean decision period clearly shows that the decision tree typically offers a two order of magnitude improvement over the optimisation. It should also be noted that the maximum decision period of the optimisation is nearly 20 seconds which is clearly impractical for a UAV given the system dynamics.

The decreased decision period is due to the decision tree making only an approximation to the optimised path while the MPC–MILP controller expends a large amount of effort to guarantee the optimal solution. Comparing the decision periods of the two algorithms it is clear that the LDT is far more suited to a real-time environment. Section 7 demonstrated that the LDT is also able to generate good approximations to the MPC–MILP controller, some summary statistics on the same test scenarios used to compare the computational complexity are now presented.

The quality of the paths can be compared by comparing the mean value of the objective function for both controllers at each time-step. Clearly, the objective function is never calculated by the decision tree controller but can be determined a posteriori by collecting state information at each time-step. Table 6 shows that in the best case the optimisation and decision tree perform equally well but that in general the optimisation slightly out-performs the decision tree. In the worst case, the cost of the decision tree trajectory is nearly twice that of the optimisation. Comparing the

Table 7
Comparison of optimisation and decision tree path length.

	Mean path length (steps)		
	Min	Mean	Max
Optimisation	9	13.5	23
Decision tree	9	12.48	24

mean path lengths (Table 7) suggests that the decision tree is more likely to take a shorter, and hence riskier, route than the optimisation although the difference is not significant.

8. Conclusions

The cloned controller has been shown to make good approximations to the optimised trajectories and to generalise well to examples consisting of significantly more obstacles than present in the training data. In addition to making a good approximation, the decision period of the LDT controller is sufficiently small to permit use in a real time system. It is noted that the obstacle avoidance problem is proposed as an example of a class of problem that challenges real-time implementation with current hardware; as hardware improves, such problems will continue to exist although the exact problems will change. Furthermore, the interpretability of the decision tree has been shown to provide useful insight into the controller's decision making.

Lookahead-LID3 was proposed and offers a further improvement over the standard LDT algorithm. The success of this method is likely to be heavily influenced by the method of training data generation. The algorithm causes the UAV to move towards regions of low uncertainty, i.e. the regions with most data points. This is an important result and suggests a powerful technique for approximating any model predictive controller and would be an interesting area for future study.

The sensitivity to the training process and underlying simulation is another area that could benefit from further study. Whilst this paper has presented a method that is capable of making very fast, accurate and interpretable predictions that are robust to uncertainty, it has not attempted to validate the simulation that it is derived from. Furthermore, demonstration in an operational environment may reveal additional challenges such as sensor noise. Given the proposed framework where the LDT provides an input to a lower-level autopilot then the extension of the simulated controller to an operational system should demonstrate good robustness to operational challenges as long as the simulation is an accurate representation of the desired mission characteristics.

References

[1] E. Morales, C. Sammut, Learning to fly by combining reinforcement learning with behavioural cloning, in: Proc. of the Intl. Conf. on Machine Learning, 2004, p. 76.
[2] C. Sammut, S. Hurst, D. Kedzier, D. Michie, Learning to fly, in: Proc. of the Intl. Conf. on Machine Learning, 1992.
[3] D. Šuc, I. Bratko, Skill modeling through symbolic reconstruction of operator's trajectories, IEEE Trans. Syst. Man Cybern., Part A, Syst. Humans 30 (2000) 617–624.
[4] D. Rathbun, B. Capozzi, An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments, in: Digital Avionics Systems Conf., 2002.
[5] A. Richards, J.P. How, Robust variable horizon model predictive control for vehicle maneuvering, Int. J. Robust Nonlinear Control 16 (7) (2006) 333–351.
[6] J. Clapper, J. Young, J. Cartwright, J. Grimes, Unmanned systems roadmap: 2007–2032, Tech. rep., U.S. Department of Defense, 2007.
[7] R. Parasuraman, V. Riley, Humans and automation: use, misuse, disuse, abuse, human factors, Hum. Factors 39 (2) (1997) 230–253.
[8] A. Goossens, Development and evaluation of level 3 situation awareness support functions for a UAV operator station, in: Digital Avionics Systems Conference, vol. 2, IEEE, 2004.
[9] C. Mencar, A. Fanelli, Interpretability constraints for fuzzy information granulation, Inf. Sci. 178 (2) (2008) 4585–4618.
[10] J.M. Alonso, L. Magdalena, G. González-Rodríguez, Looking for a good fuzzy system interpretability index: an experimental approach, Int. J. Approx. Reason. 51 (1) (2009) 115–134.
[11] A. Riid, E. Rustern, Interpretability of fuzzy systems and its application to process control, in: Proc. of the IEEE Fuzzy Systems Conference, 2007, pp. 1–6.
[12] S.-M. Zhou, J.Q. Gan, Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling, Fuzzy Sets Syst. 159 (23) (2008) 3091–3131.
[13] R. Mikut, J. Jäkel, L. Gröll, Interpretability issues in data-based learning of fuzzy systems, Fuzzy Sets Syst. 150 (2) (2005) 179–197.

[14] O. Cordon, F. Herrera, Author's reply [to comments on 'a proposal to improve the accuracy of linguistic modelling'], IEEE Trans. Fuzzy Syst. 11 (6) (2003) 866–869.
[15] E.H. Mamdani, Application of fuzzy logic to approximate reasoning using linguistic synthesis, in: Proceedings of the Sixth International Symposium on Multiple-Valued Logic, MVL '76, 1976, pp. 196–202.
[16] S. Guillaume, Designing fuzzy inference systems from data: an interpretability-oriented review, IEEE Trans. Fuzzy Syst. 9 (3) (2001) 426–443.
[17] R. Jeffrey, The Logic of Decision, University of Chicago Press, 1965.
[18] J. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
[19] L. Dubins, On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents, Am. J. Math. 79 (3) (1957) 497–516.
[20] G. Yang, V. Kapila, Optimal path planning for unmanned air vehicles with kinematic and tactical constraints, in: IEEE CDC, 2002, pp. 1301–1306.
[21] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[22] X. Yun, K. Tan, A wall-following method for escaping local minima in potential field based motion planning, in: Intl. Conf. on Advanced Robotics, 1997, pp. 421–426.
[23] O. Takahashi, R. Schilling, Motion planning in a plane using generalized Voronoi diagrams, in: Robotics and Automation, 1989, pp. 143–150.
[24] A. Stentz, Optimal and efficient path planning for partially-known environments, in: Proc. of IEEE Conf. on Robotics and Automation, 1994, pp. 3310–3317.
[25] N. Richards, M. Sharma, A hybrid A*/automaton approach to on-line path planning with obstacle avoidance, in: Intelligent Systems Technical Conf., 2004.
[26] D. Fogel, L. Fogel, Optimal routing of multiple autonomous underwater vehicles through evolutionary programming, in: Autonomous Underwater Vehicle Technology, 1990, pp. 44–47.
[27] J. Wilburn, J. Cole, M. Perhinschi, B. Wilburn, Comparison of a fuzzy logic controller to a potential field controller for real-time UAV navigation, in: AIAA Guidance, Navigation, and Control Conference, 2012.
[28] A. Richards, J. How, Aircraft trajectory planning using MILP, in: AIAA ACC, 2002.
[29] R. Bitmead, M. Gevers, V. Wertz, Adaptive Optimal Control: The Thinking Man's GPC, Prentice Hall, 1990.
[30] J. Maciejowski, Predictive Control with Constraints, Prentice Hall, England, 2002.
[31] J. Rossiter, Model-Based Predictive Control – A Practical Approach, CRC Press, Florida, 2003.
[32] B. Kouvaritakis, J. Rossiter, A. Chang, Stable generalised predictive control: an algorithm with guaranteed stability, IEE Proc. Part D, Control Theory Appl. 139 (4) (1992) 349–362.
[33] O. Turnbull, A. Richards, J. Lawry, M. Lowenberg, Fuzzy decision tree cloning of flight trajectory optimisation for rapid path planning, in: Proc. of the IEEE CDC, 2006.
[34] Z. Qin, J. Lawry, Prediction trees using linguistic modelling, in: World Congress of Intl. Fuzzy Systems Association, 2005.
[35] Z. Qin, J. Lawry, Decision tree learning with fuzzy labels, Inf. Sci. 172 (2005) 91–129.
[36] J. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[37] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[38] C. Janikow, Fuzzy decision trees: issues and methods, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 28 (1) (Feb. 1998) 1–14.
[39] Y. Peng, P. Flach, Soft discretization to enhance the continuous decision tree induction, in: ECML/PKDD Workshop: IDDM, 2001.
[40] C. Olaru, L. Wehenkel, A complete fuzzy decision tree technique, Fuzzy Sets Syst. 138 (2) (2003) 221–254.
[41] Y. Yuan, M. Shaw, Induction of fuzzy decision trees, Fuzzy Sets Syst. 69 (2) (1995) 125–139.
[42] L. Breiman, J. Friedman, C.J. Stone, R. Olshen, Classification and Regression Trees, Wadsworth Inc., 1984.
[43] J. Lawry, A framework for linguistic modelling, Artif. Intell. 155 (2004) 1–39.
[44] Z. Qin, J. Lawry, A tree structured classification model based on label semantics, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy, 2004, pp. 261–268.
[45] N. Randon, J. Lawry, A new linguistic prediction method based on random set semantics, in: Proc. of the UK Workshop on Computational Intelligence, 2003, pp. 81–88.
[46] N. Randon, J. Lawry, D. Han, I. Cluckie, River flow modelling based on fuzzy labels, in: Information Processing and Management of Uncertainty, 2004.
[47] S. Royston, J. Lawry, K. Horsburgh, A linguistic decision tree approach to predicting storm surge, Fuzzy Sets Syst. 215 (2013) 90–111.
[48] N. Randon, Fuzzy and random set based induction algorithms, Ph.D. thesis, University of Bristol, 2004.
[49] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Netw. 5 (6) (1994) 989–993.
[50] J. Lawry, D. McCulloch, N. Randon, I. Cluckie, Artificial Intelligence Techniques for Real-Time Flood Forecasting, Wiley, 2010.
[51] Z. Qin, J. Lawry, N. Randon, Prediction and query evaluation using linguistic decision trees, Appl. Soft Comput. 11 (2011) 3916–3928.
[52] J. Lawry, H. He, Linguistic decision trees for fusing tidal forecasting models, in: Combining Soft Computing and Statistical Methods in Data Analysis, vol. 77, 2010, pp. 403–410.
[53] P. Langley, H. Simon, Applications of machine learning and rule induction, Commun. ACM 38 (11) (1995) 54–64.
[54] S. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 21 (3) (1991) 660–674.
[55] U. Fayyad, P. Smyth, N. Weir, S. Djorgovski, Automated analysis and exploration of image databases: results, progress, and challenges, J. Intell. Inf. Syst. 4 (1) (1995) 7–25.
[56] N. Karba, R. Drole, Expert system for the cold rolling mill of the steel works Jesenice, in: Proc. of the Symposium on Information Technologies, Sarajevo, 1990.
[57] D. Michie, Directions in machine intelligence, Comput. Bull. 4 (4) (1992) 9–11.

[58] J. Baldwin, T. Martin, B. Pilsworth, Fril-Fuzzy and Evidential Reasoning in Artificial Intelligence, John Wiley & Sons, Inc., New York, NY, 1995.
[59] E.H. Ruspini, A new approach to clustering, Inf. Control 15 (1) (1969) 22–32.
[60] J. Platts, E. Berglund, M. Hagström, P. Ögren, I. Panella, S. Howell, A. McCallum, Autonomy in UAVs: design challenge, Tech. rep., Group for Aeronautical Research and Technology in Europe (GARTEUR), Flight Mechanics Action Group 14 (FM AG14), November 2003.
[61] R. Beard, T. McLain, M. Goodrich, E. Anderson, Coordinated target assignment and intercept for unmanned air vehicles, Robot. Autom. 18 (6) (2002) 911–922.
[62] W. Kamal, D. Gu, I. Postlethwaite, Real-time trajectory planning for UAVs using milp, in: Proceedings of the Joint IEEE Conference on Decision and Control/European Control Conference, 2005.
[63] F. Borrelli, D. Subramanian, A. Raghunathan, L. Biegler, MILP and NLP techniques for centralized trajectory planning of multiple unmanned vehicles, in: Proceedings of the American Control Conference, 2006.
[64] T. Schouwenaars, M. Valenti, E. Feron, J. How, Implementation and flight test results of MILP-based UAV guidance, in: IEEE Aerospace Conference, 2005.
[65] IBM, Ibm ilog cplex website [online], March 2012.
[66] T. Schouwenaars, E. Feron, J. How, Receding horizon path planning with implicit safety guarantees, in: Proceedings of the American Control Conference, 2004.
[67] M. Earl, R. D'Andrea, Iterative milp methods for vehicle-control problems, IEEE Trans. Robot. 21 (2005) 1158–1167.
[68] L. Mariner, Cleared for Takeoff: English for Pilots, AE Link Publications Incorporated, 2007.
[69] O. Turnbull, Linguistic decision tree cloning of optimised trajectories for real-time obstacle avoidance, Ph.D. thesis, University of Bristol, 2008.
[70] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 27:1–27:27, software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[71] L. Breiman, J. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth International Group, 1984.
[72] Z. Qin, J. Lawry, Decision tree learning with fuzzy labels, Inf. Sci. 172 (2005) 91–129.
[73] M.J. Gacto, R. Alcalá, F. Herrera, Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems, IEEE Trans. Fuzzy Syst. 18 (3) (2010) 515–531.
[74] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall, Upper Saddle River, 1996.
[75] J.M. Alonso, L. Magdalena, Special issue on interpretable fuzzy systems, Inf. Sci. 181 (20) (2011) 4331–4339.
[76] M.J. Gacto, R. Alcalá, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures, Inf. Sci. 181 (20) (2011) 4340–4360.
[77] G.A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information, Psychol. Rev. 63 (2) (1956) 81.